

**CLAIMS**

1. A method of executing programs on a non-native platform, including the step of:
  - i) executing a plurality of programs in two or more software emulators;wherein during the execution of the programs at least one program monitors or controls at least one other program's threads or processes using an interface.
2. A method as claimed in claim 1 wherein each program executes within a separate software emulator.
3. A method as claimed in claim 2 wherein the interface is between the software emulator of the monitoring/controlling program and the software emulator of the monitored/controlled program.
4. A method as claimed in claim 3 wherein each software emulator is emulating the same platform.
5. A method as claimed in claim 4 wherein all the software emulators are executing on a single computer system.
6. A method as claimed in claim 5 wherein the computer system is UNIX-based.
7. A method as claimed in claim 5 wherein each software emulator is a dynamic translation software emulator.
8. A method as claimed in claim 1 wherein the monitoring/controlling program is a debugging program and the monitored/controlled program is a program to be debugged.
9. A method as claimed in claim 8 wherein the debugging program is a gdb-based debugger.
10. A method as claimed in claim 1 wherein the interface includes:
  - i) a first module which interfaces with the software emulator of the monitoring/controlling program;

- ii) a second module which interfaces with the software emulator of the monitored/controlled program; and
- iii) a framework through which the first and second module communicate.

- 5      11. A method as claimed in claim 10 wherein the framework is an inter-process data exchange mechanism.
12. A method as claimed in claim 11 wherein the inter-process data exchange mechanism is an inter-process communications primitive.
- 10      13. A method as claimed in claim 12 wherein the inter-process communications primitive is any one selected from the set of pipe, socket, and shared memory area.
- 15      14. A method as claimed in claim 10 wherein the second module includes a thread which polls for requests received through the framework and services the requests when they arrive.
15. A method as claimed in claim 10 wherein the first module processes trace and trace-wait system calls made by the monitoring/controlling program.
- 20      16. A method as claimed in claim 10 wherein the second module services requests received from the first module through the framework.
- 25      17. A method as claimed in claim 1 wherein each software emulator intercepts each entry into OS mode made by the emulated program and notifies the interface.
18. A method as claimed in claim 1 wherein the software emulator of the monitoring/controlling program and the software emulator of the monitored/controlled program execute on different computer systems.
- 30      19. A method as claimed in claim 1 wherein the monitoring/controlling program is a tracing program.
- 35      20. A method as claimed in claim 1 wherein the monitoring/controlling includes the use of trace and trace-wait system calls.

21. A system for executing programs on a non-native platform including:

- 5
- i) a first software emulator adapted to execute a first program, to intercept calls from the first program to monitor or control the processes or threads of a second program, and to transmit the calls to an interface system;
  - ii) a second software emulator adapted to execute the second program, to receive the calls from the interface system, and to effect the calls on the processes or threads of the second program; and
  - 10 iii) an interface system adapted to receive the calls from the first software emulator and to transmit the calls to the second software emulator.

22. A system as claimed in claim 21 wherein the second software emulator is further adapted to intercept responses to the calls from the second program and to transmit the responses to the interface system, the interface system is further adapted to receive the responses from the second software emulator and transmit the responses to the first software emulator, and the first software emulator is further adapted to receive the responses and to send the responses to the first program.

20

23. A system as claimed in claim 22 wherein the first program is a debugging program.

24. A method of debugging a program on a non-native platform, including the steps of:

25

- i) executing a debugging program on a first software emulator;
- ii) executing the program on a second software emulator;
- iii) the debugging program making calls to trace into processes or threads of the program; and
- 30 iv) transmitting the calls using an interface from the first software emulator to the second software emulator.

25. Software for effecting the method of claim 1.

35

26. Storage media containing software as claimed in claim 25.

27. A computer system for effecting the method of claim 1.

28. A program debugged by the method of claim 24.